Visual Testing with PyCharm and pytest

Brian Okken & Paul Everitt

May 11, 2018

Who we are

Paul Everitt

PyCharm and WebStorm Developer Advocate at JetBrains

Brian Okken

Author of “Python Testing with pytest”
Host of “Test & Code” podcast
Co-Host of “Python Bytes” podcast
Lead Software Engineer at Rohde & Schwarz

Testing

Testing ...

- is necessary, but not my favorite
- takes valuable time away from development
- helps me spend more time engineering and refactoring
- helps us ship faster
- gives me confidence to experiment and play without fear
- is freaking awesome (may just be my hand at this point)

Let’s test something

The cards project

```
$ cards list
  ID   owner    done    summary
----  -------  ------  ----------------------
   1   okken            Prepare for PyCon talk
   2   okken    x       Order stickers
   3   okken    x       Order business cards
   4   okken            pack
```

oh, CRUD!

```
$ cards --help
Usage: cards [OPTIONS] COMMAND [ARGS]...

...

Commands:
  add      add a card
  count    list count
  delete   delete a card
  list     list cards
  update   update card
```

but first ...

for those of you following along at home

```
$ git clone https://github.com/okken/cards.git
$ cd cards
$ python3.6 -m venv venv --prompt cards
$ source venv/bin/activate  # venv\Scripts\activate  (if
Windows)
(cards) $ pip install -e .
(cards) $ pip install -r requirements_dev.txt
```

let’s run some tests

```
(cards) $ pytest
================== test session starts ===================
collected 40 items

tests/api/test_alac.py .                                  [  2%]
tests/api/test_list_filter.py ...................         [ 50%]
tests/api/test_tracer_bullets.py .......                  [ 67%]
tests/cli/test_alac.py .                                  [ 70%]
tests/cli/test_list_format.py ......                      [ 85%]
tests/cli/test_tracer_bullets.py ......                   [100%]

=============== 40 passed in 0.62 seconds ================
```

maybe just the cli tests

```
(cards) $ pytest tests/cli
================== test session starts ===================
collected 13 items

tests/cli/test_alac.py .                           [  7%]
tests/cli/test_list_format.py ......               [ 53%]
tests/cli/test_tracer_bullets.py ......            [100%]

=============== 13 passed in 0.23 seconds ================
```

or just one file

```
(cards) $ pytest -v tests/cli/test_tracer_bullets.py
===================== test session starts ====================
collected 6 items

tests/cli/test_tracer_bullets.py::test_add PASSED         [ 16%]
tests/cli/test_tracer_bullets.py::test_list PASSED        [ 33%]
tests/cli/test_tracer_bullets.py::test_list_filter PASSED [ 50%]
tests/cli/test_tracer_bullets.py::test_count PASSED       [ 66%]
tests/cli/test_tracer_bullets.py::test_update PASSED      [ 83%]
tests/cli/test_tracer_bullets.py::test_delete PASSED      [100%]

================= 6 passed in 0.12 seconds ==================
```

one single test

```
(cards) $ pytest -v tests/cli/test_tracer_bullets.py::test_count
=================== test session starts ===================
collected 1 item

tests/cli/test_tracer_bullets.py::test_count PASSED  [100%]

================ 1 passed in 0.04 seconds =================
```

or just

```
(cards) $ pytest -v -k test_count
================== test session starts ====================
collected 40 items / 38 deselected

tests/api/test_tracer_bullets.py::test_count PASSED  [ 50%]
tests/cli/test_tracer_bullets.py::test_count PASSED  [100%]

======== 2 passed, 38 deselected in 0.17 seconds =========
```

maybe more specific

```
(cards) $ pytest -v -k 'cli and test_count'
=================== test session starts ====================
collected 40 items / 39 deselected

tests/cli/test_tracer_bullets.py::test_count PASSED   [100%]

========= 1 passed, 39 deselected in 0.19 seconds ==========
```

Some other flags for picking which tests to run, and when to stop.

```
(cards) $ pytest --help
...
  -m MARKEXPR             run tests w/ given mark expression
  -x, --exitfirst         exit on first error/failed test
  --maxfail=num           exit after first num failures/errors.
  --lf, --last-failed     rerun failed tests from last run
  --ff, --failed-first    same, but then run the rest
  --nf, --new-first       run new test files, then the rest
```

enter PyCharm

PyCharm + pytest

= awesome

you can …

- run different levels of tests, just like the command line
- pass in any wonderful pytest flag you can’t live without
- debug tests and code at the same time
- (even without a failing test, this is fun to do, we can just put a breakpoint in a test and do that)
- use a REPL pre-loaded with local variables right in the middle of a test
- run coverage reports

PyCharm + pytest setup

(is this necessary, or should we push this to supporting material?) if we cull it, we should include it in some appendix slides at the end.

some random notes

an early version of intro notes

What we want to show you today is how you can use pytest and PyCharm to make you a more productive developer. Realy, we want to show you that using pytest and PyCharm can help you: * maximize the effectiveness of your time spent writing and running tests * speed up your first implementation * allow you the most time possible for engineering

regarding TDD

I have some notes on TDD, but it seemed forced to put them in the middle before just jumping in. We can talk about it at the end, or just not worry about it.

TDD refresher

- Red
- Green
- Refactor

TDD according to Brian

- Writing Tests
- First Implementation
- Making the code something you are proud of

don’t forget a “for more info” slide

to discuss at the end

This talk has slides at the end with reference material so you don’t have to write notes, even during the demo part. Also, Brian’s easy to find in the podcast booth for much of tomorrow. And we will hold an open session tomorrow or Sunday to be availabel for questions.